Introduction to Parallel Computing

Dr. John E. Dorband
NASA
July 13, 2004

Computing

- Computation

  What is the purpose of a Computer?

  To Compute FAST

  What Limits the speed of a computer?

  Component Speed (gates)

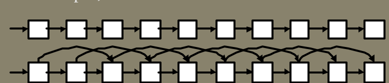
  Distance between components (speed of light)

Parallel Programming

How do you design a parallel program?

Parallel programming is NOT the dividing up of a serial program

For example, find the end of a list:



Essential aspects of Parallel Programming

- Computation
- Communication

Simple Parallelism

- Mandlebrot set
- SETI online
- Ensembles
  - Climate model
  - Solar atmosphere
- The Web
- Data grid

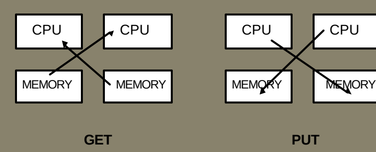
Complex Parallel

- Image generation
- Unstructured Grid computations
- Rectangular Grid computations
- Image and signal processing
- FFT – Fast Fourier Transforms
- Cellular Automata

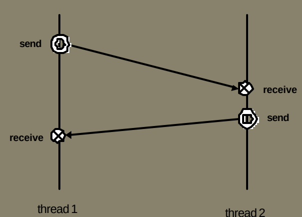
Communication

- Get/Put
- Send/Receive
- Scatter/Gather
- All to All
- Broadcast/Reduction
- Scan

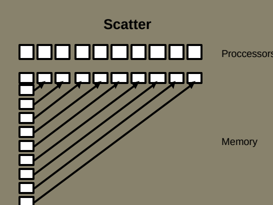
Get / Put



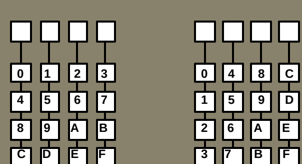
Send / Receive



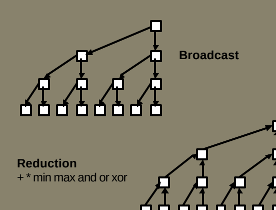
Scatter / Gather



All to All



Broadcast / Reduction



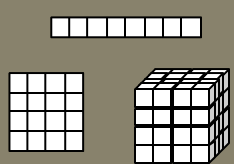
Scan

Scan add	1	1	1	0	1	0	1	1
	1	2	3	3	4	4	5	6
Scan xor	1	1	1	0	1	0	1	1
	1	0	1	1	0	0	1	0

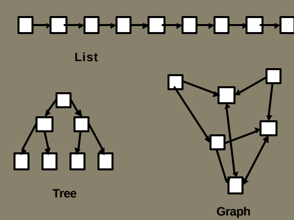
Topologies

- Arrays
- Lists / Graphs
- Hyper-cubes

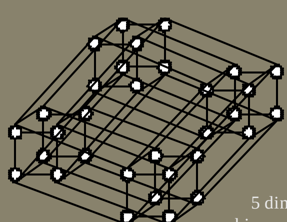
Arrays



Lists / Graphs



Hyper-Cubes

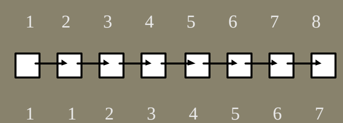


5 dimensional binary hyper-cube

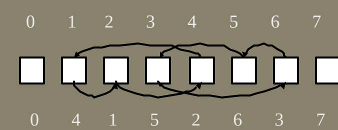
Permutations

- Shift
- Shuffle
- Transpose

Shift



Shuffle



Transpose

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

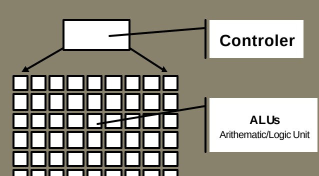
0	4	8	C
1	5	9	D
2	6	A	E
3	7	B	F

Parallel Architectures

- SIMD
- Pipeline
- MIMD
  - SMP / shared
  - NUMA / distributive shared
  - Clusters / distributive

SIMD

single-instruction stream multiple-data stream



Solomon Computer

9 bit serial processors in 1962

NASA/Goodyear MPP

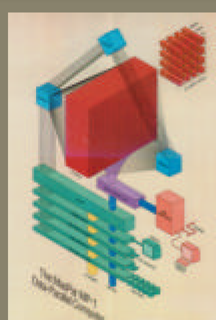
16,384 proc

1024 bits/proc

MasPar MP1/MP2

1990-2000
16K custom proc.
$1.5-2 Million
6.2 GFlop
3.7 sustained (PPM)

SIMD architecture



Present Day SIMD

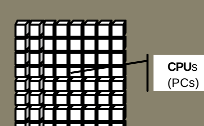
- Pentium MMX
- FPGA architectures

Pipeline

- Numeric pipeline
  - Cray
  - Multiplier
- Functional pipeline
  - Pentium /Athlon

MIMD

multiple-instruction stream multiple-data stream



Beowulf I: Wiglaf

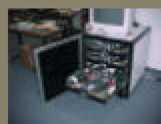
1994

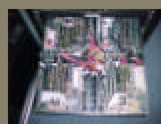
16 - 100Mhz 486DX4

$40,000

50 MFlops sustained (PPM)

Dual 10Mhz ethernet





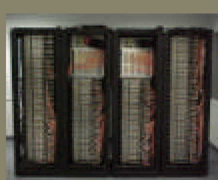
theHIVE

Highly-Parallel Integrate Virtual Environment

1997	1999
128 - 200Mhz Pentium Pro	128 - 200Mhz Pentium Pro
$210,000	100Mhz switched ethernet
7.5 Gflops sustained (PPM)	72 port switch
100Mhz switched ethernet	

Thunderhead

Fall 2002



2.4 Ghz P4 Xeon cluster
- 1 console node
- 3 host nodes
- 256 dual P4 compute nodes
- 8 dual P4 spare nodes
- Fast ethernet
- 2 Ghz Myrinet

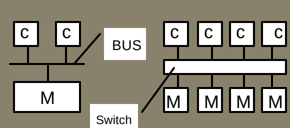
106th on the TOP500 List Nov. 2003

Characteristics:
- peak of 2.5 TF
- 1.1Tflop on LINPAK benchmark
- 20 TB scratch storage
- 12.8 TB raided storage

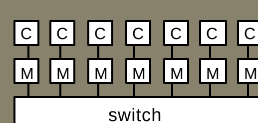
MIMD Architectures

- SMP / shared
- NUMA / distributive shared
- Clusters / distributive

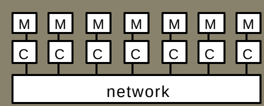
SMP
symmetric multi-processor



NUMA
non-uniform memory access



Cluster distributed memory



Parallel Computing Issues

- Communication
  - Bandwidth
  - Latency
  - Buffering
- Computation
  - Determinism
  - Race Conditions
  - Dead lock

Bandwidth

- Lower bandwidth
  → higher communication time

Latency

- 1. The portion of the communication time not proportional to the message size.
- 2. The time it takes to send a message.
  - Latency hiding

Buffering Issues

- Memory Copying
  - Costly in any architecture
- Double buffering
  - When viable double buffering can be used to hide communication cost by allowing computations to be performed during communication.

Determinism

- Types
  - Non-deterministic
  - Deterministic
  - Ordered
- Effected Operations
  - Floating point Multiply and Add
  - Results consistency and reproducibility
  - Debugging

Race Condition

- Read buffer before data was written to it.
- Writing to a buffer before it has been read
- Writing to variables in the wrong order

Deadlock

- Two or more processes requesting the same set of resources that are then distributed among processes
- Two processes sending to each other, causing both to wait for the other to receive

Synchronize

- Barriers
- Blocking/Non-blocking

Barrier

- A point during execution where all relevant processes or threads must wait for each other.

Blocking / Non-Blocking

- Blocking communications
  - Communications that does not return until it has completed
- Non-blocking communications
  - Communication operation that returns before it is complete
  - Allows computation to be performed while operation is being performed

Non-synchronize

- Mutual Exclusion (critical sections)
  - Monitor
  - Semaphores (P/V)
  - Test and Set

Monitor

- Begin monitor
    - <critical section>
- End monitor

PV Semaphore

- *Dijkstra – counting semaphore*
- *Proberen te verlangen* (wait) [P]
    - If semaphore>0, perform critical section
    - If semaphore=<0, wait until semaphore>0
    - decrement semaphore
- *verhogen* (post) to increase a semaphore [V]
    - increment semaphore

PV Semaphore

- declare sema4
- det sema4=1
- P(sema4)
    - <critical section>
- V(sema4)

Binary Semaphore

- declare sema4
- lock(sema4)
    - <critical section>
- unlock(sema4)

Test and Set

- Atomic operation
    - Test variable
    - Set variable to TRUE
- Restrict access during:
    - interruptible code
    - multi-processor shared memory system

Performance Metrics

- Floating point operations per sec (flops)
- Time to completion
- Speed up
- Efficiency

FLOPS

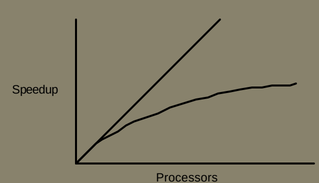
- Why is flops a legitimate metric
  - Usually floating point values and operations represent work necessary to solve problem
  - Usually integer operations represent the control of an algorithm
  - The solution of a problem is not necessarily dependent on a specific algorithm

Time to completion

- Why is "time to completion" not a good metric.
  - It is little use to any one but the writer of the algorithm.
  - It depends on a fixed size problem.

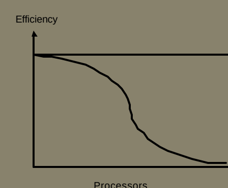
Speed Up

$$Speedup = \frac{Time\_for\_1}{Time\_for\_n}$$



Efficiency

$$Efficiency = \frac{Speedup}{N}$$



Performance Issues

- Amdahl's Law
- Latency/Bandwidth
- Computation/Communication

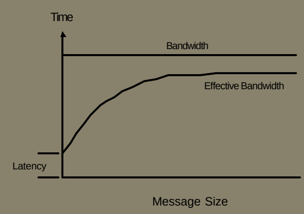
Amdahl's Law

$$T_N = \frac{(1-f)T_1}{N} + f\,T_1$$

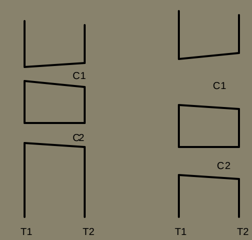
$$Speedup = \frac{N}{(1-f)+Nf} \le \frac{1}{f}$$

$$Efficiency = \frac{1}{(1-f)+Nf} \le \frac{1}{fN}$$

Latency / Bandwidth



Computation/Communication



Final words of Wisdom

!! Think Parallel !!

Don’t Give Up

Questions?